UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/977,715	10/12/2001	David S. Allison	36159/098001; P5944	3306

32615 7590 03/30/2011

OSHA LIANG LLP/Oracle
TWO HOUSTON CENTER
909 FANNIN, SUITE 3500
HOUSTON, TX 77010

EXAMINER	
ZHEN, LI B	

ART UNIT	PAPER NUMBER
2197	

NOTIFICATION DATE	DELIVERY MODE
03/30/2011	ELECTRONIC

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Notice of the Office communication was sent electronically on above-indicated "Notification Date" to the following e-mail address(es):

docketing@oshaliang.com
lord@oshaliang.com
hathaway@oshaliang.com

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE BOARD OF PATENT APPEALS
AND INTERFERENCES

Ex parte DAVID S. ALLISON

Appeal 2009-009962
Application 09/977,715
Technology Center 2100

Before JAY P. LUCAS, JOHN A. JEFFERY, and
ST. JOHN COURTENAY III, *Administrative Patent Judges*.

JEFFERY, *Administrative Patent Judge*.

DECISION ON APPEAL

Appellant appeals under 35 U.S.C. § 134(a) from the Examiner's rejection of claims 23, 26-30, and 33-40. We have jurisdiction under 35 U.S.C. § 6(b). We affirm.

STATEMENT OF THE CASE

Appellant's invention communicates to control threads through streams. *See generally* Spec. 2; Abstract. Claim 23 is illustrative with a key disputed limitation emphasized:

> 23. A method for communicating between threads, comprising:
> invoking a first thread;

associating a first input stream and a first output stream with the first thread;

invoking a second thread;

associating a second input stream and a second output stream with the second thread;

invoking a stream operator to write a first data value from the first thread to the second thread, wherein the stream operator connects the first output stream to the second input stream and sends the first data value from the first output stream to the second input stream;

using the second thread to generate a second data value by performing an operation on the first data value; and

invoking the stream operator to write the second data value from the second thread to the first thread, wherein the stream operator connects the second output stream to the first input stream and sends the second data value from the second output stream to the first input stream,

wherein at least one selected from the group consisting of the first thread and the second thread manages an operating system process and comprises:

a program counter;

a stack;

a state; and

a register set.

The Examiner relies on the following as evidence of unpatentability:

Tyler	US 6,131,183	Oct. 10, 2000
Carlson	US 6,842,898 B1	Jan. 11, 2005 (filed June 10, 1999)

THE REJECTION

The Examiner rejected claims 23, 26-30, and 33-40 under 35 U.S.C. § 103(a) as unpatentable over Tyler and Carlson. Ans. 3-5.[1]

[1] Throughout this opinion, we refer to (1) the Appeal Brief filed October 20, 2008; (2) the Examiner's Answer mailed January 8, 2009; and (3) the Reply Brief filed March 9, 2009.

CONTENTIONS

Regarding representative claim 23, the Examiner finds that Tyler discloses a method for communicating between first and second "threads," which the Examiner equates to (1) Tyler's ARC/INFO[2] controller 44 (and its corresponding main routine 62 and "readStdin" subroutine 60), and (2) ARC/INFO child program 28 (and corresponding "XtMainLoop" routine 28a), respectively. Ans. 3-4, 7-8. The Examiner also notes that these "threads" each have corresponding input and output streams designated as "stdin" and "stdout," respectively. Ans. 4, 7.

According to the Examiner, when the main routine of Tyler's ARC/INFO controller registers its "readStdin" subroutine and associated callback parameters such that this subroutine can be called back from the ARC/INFO child program's "XtMainLoop" routine, the "first thread" (i.e., the ARC/INFO controller 44) is said to write a first data value to the "second thread" (ARC/INFO child program 28) as claimed to facilitate this callback. Ans. 7-8. Tyler is also said to write a second data value from the "second thread" to the "first thread" when the ARC/INFO child program's "XtMainLoop" routine calls the "readStdin" subroutine in the "first thread" based on the registered callback parameters. Ans. 8.

Based on this analysis, the Examiner finds that Tyler discloses every recited feature of claim 23 except for the threads comprising at least one of a program counter, stack, state, and a register set, but cites Carlson as teaching these features in concluding the claim would have been obvious. Ans. 5.

[2] ARC/INFO® is a commercially-available mapping and geographic information system computer software program. Tyler, col. 1, ll. 16-22.

Appellant argues that the cited prior art does not teach or suggest (1) using a second thread to generate a second data value by performing an operation on the first data value, and (2) writing the second data value from the second thread to the first thread using the second thread's output stream as claimed. App. Br. 8-12; Reply Br. 4-5. According to Appellant, Tyler's second data value (i.e., the call) is generated responsive to a Unix "TTY" command from a terminal emulator—not by performing an operation on Tyler's first data value as claimed. Reply Br. 4-5.

The issue before us, then, is as follows:

ISSUE

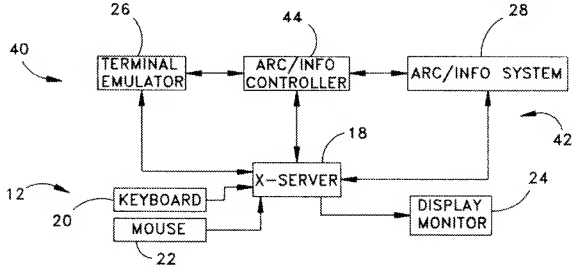
Under § 103, has the Examiner erred in rejecting claim 23 by finding that Tyler and Carlson collectively would have taught or suggested (1) using a second thread to generate a second data value by performing an operation on a first data value, and (2) writing the second data value from the second thread to the first thread using the second thread's output stream as claimed?

FINDINGS OF FACT (FF)

1. Tyler's system controls graphic user interface (GUI) and command line (TTY) operations without manually switching between these modes. To this end, Tyler's computer system 40 includes an ARC/INFO controller 44 that (1) is located between a terminal emulator 26 and ARC/INFO system (program) 28,[3] and (2) communicates with X-Server 18. Tyler, Title;

[3] Tyler uses the labels "ARC/INFO system" and "ARC/INFO program" interchangeably to denote numeral 28. *Compare* Tyler, col. 4, ll. 9-10 *with*

Abstract; col. 1, ll. 8-13; col. 4, ll. 3-12; Fig. 3. Tyler's computer system with the ARC/INFO software program and controller is shown in Figure 3 reproduced below:



Tyler's computer system with ARC/INFO program and controller in Figure 3

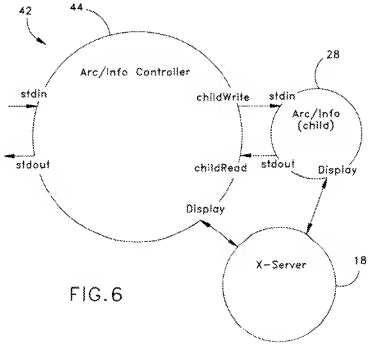
2. Tyler's ARC/INFO controller 44 and program 28 are provided with two TTY data streams: (1) a standard input "stdin," and (2) a standard output "stdout." Tyler, col. 4, ll. 49-55; Fig. 5.

3. Tyler's ARC/INFO controller 44 (1) spawns ARC/INFO program 28 as a child process, and (2) glues its "stdin" and "stdout" to "childWrite" and "childRead" which can be a pty/tty pair or set of pipes. In addition to scheduling the ARC/INFO program 28, controller 44 (1) passes its "stdin" through its "childWrite" to the ARC/INFO program's "stdin," and (2) acquires the ARC/INFO program's "stdout" through the controller's "childRead," and passes it to the controller's "stdout." Tyler, col. 4, l. 66 –

---

col. 4, l. 14. Nevertheless, we refer to Tyler's numeral 28 in this opinion as an "ARC/INFO program" for clarity and consistency.

col. 5, l. 10; Fig. 6. Data flow in Tyler's ARC/INFO controller is shown in Figure 6 reproduced below:



Data flow in Tyler's ARC/INFO controller in Figure 6

4. Tyler's ARC/INFO controller 44 performs the "readStdin" transform when data is available on its "stdin" stream. This transform (1) "wakes up" the ARC/INFO child (i.e., ensures the ARC/INFO program's attention is directed to its "stdin"); (2) relays "stdin" to "childWrite"; and (3) puts the ARC/INFO child to "sleep" (i.e., directs the ARC/INFO program's attention away from its "stdin"). Tyler, col. 5, ll. 11-41, 47-49; Figs. 7-8.

5. Tyler's Figure 8 charts the structure of (1) ARC/INFO controller 44, and (2) the ARC/INFO program 28's main routine (i.e., the "XtMainLoop" routine 28a). Controller 44 includes a "readStdin" subroutine 60 that is called by the "XtMainLoop" routine 28a in accordance with callback parameters set up by a main routine 62. Tyler, col. 3, ll. 57-

58; col. 5, ll. 42-49; Fig. 8. Tyler's Figure 8 charting the structure of (1) ARC/INFO controller 44, and (2) the ARC/INFO program's "XtMainLoop" routine is reproduced below:

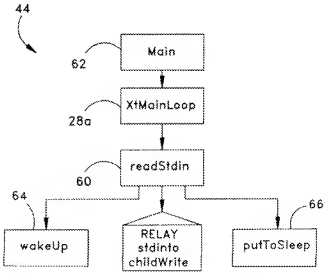


FIG.8

Tyler's Figure 8 charting the structure of the ARC/INFO controller and the ARC/INFO program's "XtMainLoop" routine

6. Tyler's ARC/INFO controller's main routine creates the ARC/INFO child process with TTY links to its "stdin" and "stdout." The "readStdin" subroutine 60 is registered so that it will be called back from within the "XtMainLoop" routine 28a when there is something to read from the ARC/INFO controller's "stdin." Tyler, col. 5, ll. 51-59; Figs. 6, 8-9.

7. In response to a TTY command from terminal emulator 26 or from an auxiliary user program, ARC/INFO program's "XtMainLoop" routine 28a (1) detects the associated event, and (2) calls the "readStdin" subroutine 60 which then calls wakeup subroutine 64. Then, the "readStdin" subroutine causes the TTY command to be relayed from the terminal emulator or user program through "childWrite" to the ARC/INFO program's "stdin." When

the operation completes, the "readStdin" subroutine (1) calls the "putToSleep" subroutine 66, and then (2) returns to the "XtMainLoop" routine. Tyler, col. 5, l. 50 – col. 6, l. 38; Figs. 6, 8-12.

ANALYSIS

The dispute before us hinges on the Examiner's interpretation of Tyler regarding (1) using a second thread to generate a second data value by performing an operation on a first data value, and (2) writing the second data value from the second thread to the first thread using the second thread's output stream as claimed. Since the Examiner's findings regarding Carlson (Ans. 5) are undisputed, we therefore confine our discussion to Tyler.

The Examiner equates Tyler's ARC/INFO controller 44 and its corresponding main routine 62 and "readStdin" subroutine 60 to the recited "first thread." Ans. 3-4, 7-8. The Examiner also equates Tyler's ARC/INFO program 28 and its corresponding "XtMainLoop" routine 28a to the recited "second thread." *Id.* This mapping is undisputed, and indeed reasonable in light of the threads' disclosed relationship and functionality. *See* FF 1-5.

The threshold question before us, then, is whether this "second thread," namely Tyler's ARC/INFO program 28, is used to generate a second data value by performing an operation on a first data value as claimed.

Regarding the recited first data value, the Examiner relies principally on the main routine of Tyler's ARC/INFO controller registering its "readStdin" subroutine and associated callback parameters such that this subroutine can be called back from the ARC/INFO child program's

"XtMainLoop" routine. Ans. 7-8; FF 3-6. That is, Tyler's "first thread" (ARC/INFO controller 44) is said to write a first data value to the "second thread" (ARC/INFO child program 28) to facilitate this callback. Ans. 7-8. We see no error in this reasoning, since Tyler at least suggests that some sort of data value would be written from the ARC/INFO controller to the ARC/INFO child program to, at a minimum, identify the data and functions associated with the respective routines to (1) facilitate the calls between these entities, and (2) execute their functions. *See* FF 3-7.

Likewise, we see no reason why Tyler's "second thread" (ARC/INFO child program 28) would not generate some sort of data value by operating on the first data value associated with the ARC/INFO controller and its registered "readStdin" subroutine to facilitate callback in accordance with the callback parameters. *See id.* Here again, skilled artisans would recognize that the second thread would be used to perform some sort of operation on the first data value (e.g., receiving or identifying that data value) to generate and write a second data value to the first thread to, at a minimum, identify the data and functions associated with the respective routines to (1) facilitate the calls between the threads, and (2) execute their functions. *See* Ans. 8; FF 3-7. We reach this conclusion noting that nothing in claim 32 precludes Tyler's respective "data values" given their scope and breadth, let alone require that they be different. Nor does claim 32 preclude the mere receipt or identification of the first data value by Tyler's ARC/INFO program as performing an operation on that value given the scope and breadth of the limitation.

Although Tyler's calls are ultimately responsive to Unix TTY commands as Appellant argues (Reply Br. 4-5; FF 1, 7), this initial

triggering of Tyler's thread-based functionality does not obviate the subsequent generation and exchange of data associated with Tyler's ARC/INFO controller and program noted above. *See* Ans. 7-8; FF 1-7. In short, Appellant's arguments (App. Br. 8-12; Reply Br. 4-5) are simply not commensurate with the scope of the claim and, in any event, do not persuasively show error in the Examiner's reliance on the exchange of data between Tyler's ARC/INFO controller and program noted above. *See* Ans. 7-8; FF 1-7.

We are therefore not persuaded that the Examiner erred in rejecting representative claim 23, and claims 26-30 and 33-40 not separately argued with particularity.

CONCLUSION

The Examiner did not err in rejecting claims 23, 26-30, and 33-40 under § 103.

ORDER

The Examiner's decision rejecting claims 23, 26-30, and 33-40 is affirmed.

No time period for taking any subsequent action in connection with this appeal may be extended under 37 C.F.R. § 1.136(a)(1)(iv).

AFFIRMED

rwk